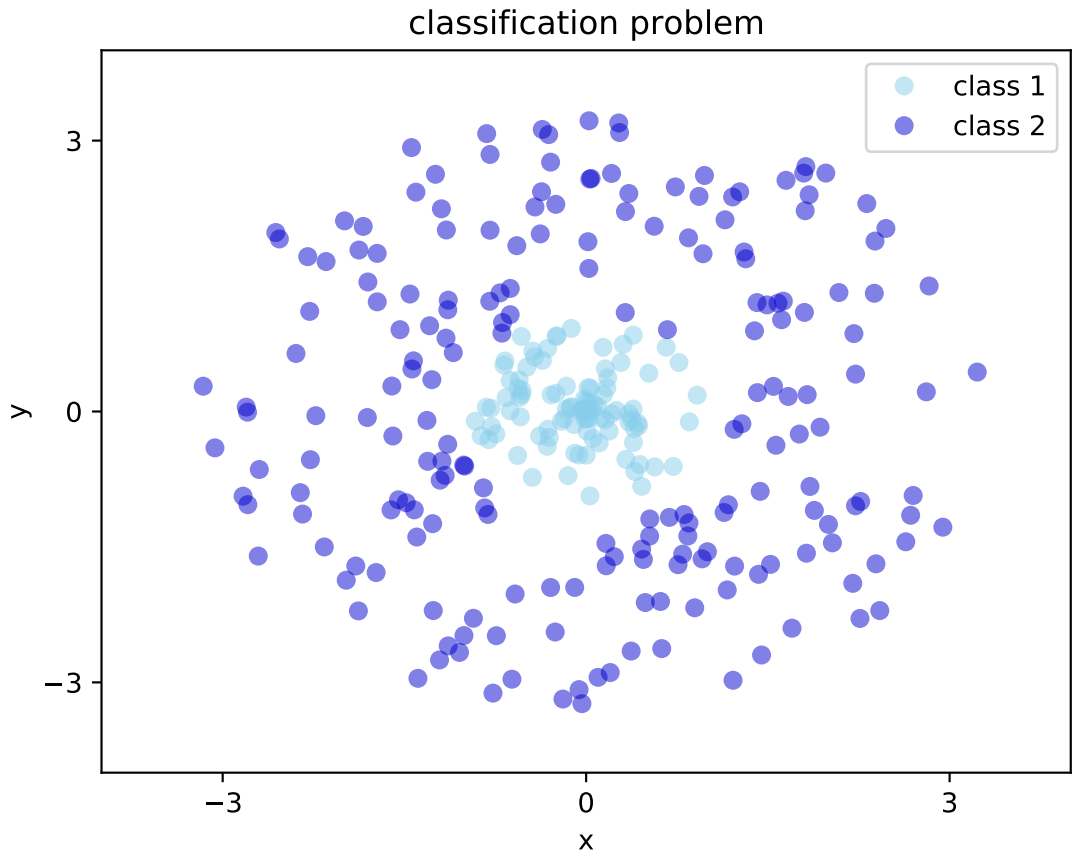
Fondamentaux théoriques du machine learning



Overview of lecture 5

Feature maps

Scoring, multiclass problems, cross validation

- Regression
- Binary classification
- Multi-class classification
- Cross-validation

Clustering

Support vector machines

- Linear separation
- Optimization problem
- Link with empirical risk minimization

Feature maps

Scoring, multiclass problems, cross validation
- Regression
- Binary classification
- Multi-class classification
- Cross-validation

Clustering

Support vector machines
- Linear separation
- Optimization problem
- Link with empirical risk minimization

Feature maps

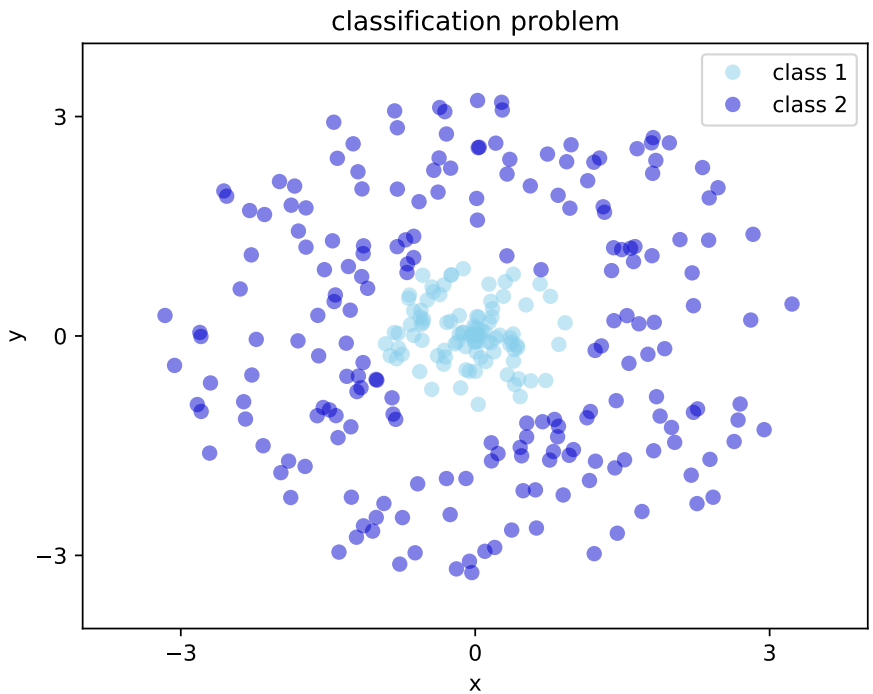
Often, we do not work with the $x_i \in \mathcal{X}$, but with **representations** $\phi(x_i)$, with $\phi : \mathcal{X} \to \mathbb{R}^d$. Possible motivations :

- $\mathcal{X}$ need not be a vector space.
- $\phi(x)$ can provide more useful **features** for the considered problem (classification, regression).
- The prediction function is then allowed to depend **non-linearly** on x. (But there can still be linear dependence in $\phi(x)$, this will often be the case).

Feature map

Exercice 1 : **Finding a feature map**

What feature map could be used to be able to **linearly separate** these two classes ?



Application to OLS and ridge

Instead of

$$X = \begin{pmatrix} x_1^T \\ \dots \\ x_i^T \\ \dots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11}, \dots, x_{1j}, \dots x_{1d} \\ \dots \\ x_{i1}, \dots, x_{ij}, \dots x_{id} \\ \dots \\ x_{n1}, \dots, x_{nj}, \dots x_{nd} \end{pmatrix}$$

The design matrix is

$$\phi = \begin{pmatrix} \phi(x_1)^T \\ \dots \\ \phi(x_i)^T \\ \dots \\ \phi(x_n)^T \end{pmatrix}$$

Application to OLS and ridge

The statistical results are maintained, as a function of d, the dimension of $\phi(x)$.

Linear estimator

We often encounter estimators of the form

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{1}$$

- They are often called "linear models"
- Being linear in θ is not the same as being linear in x.

Linear estimator

We often encounter estimators of the form

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{2}$$

- regression : $h = Id$
- classification : $h = \text{sign}$.

Linear estimator

Interpretation of a linear model as a vote, in the case of classification.

$$f(x) = h(\langle \phi(x), \theta \rangle) = h(\phi(x)^T \theta) \tag{3}$$

Kernel methods

The topic of feature maps is very rich and important in machine learning

- **kernel methods** : ϕ is **chosen**. Many famous choices are available (gaussian kernels, polynomial kernels, etc).
- **neural networks** : ϕ is **learned**.

We will have dedicated exercises on both these methods.

Scoring

Many possibilities are available to evaluate the quality of an estimator.

Regression

- $\mathcal{X} = \mathbb{R}^d$
- $\mathcal{Y} = \mathbb{R}$.

Until now we used the squared error or the mean squared error (MSE) :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{pred,i} - y_{label,i})^2 \tag{4}$$

Without normalisation, it is also called **residual sum of squares** (RSS)

$$RSS = \sum_{i=1}^{n}(y_{pred,i} - y_{label,i})^2 \tag{5}$$

Coefficient of determination

Also called $R2$. $R2 \leq 1$. We introduce the Total sum of squares (TSS)

$$TSS = \sum_{i=1}^{n} (y_{label,i} - \bar{y})^2 \tag{6}$$

where $\bar{y}$ is the mean of the labels :

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_{label,i} \tag{7}$$

Then

$$R2 = 1 - \frac{RSS}{TSS} \tag{8}$$

Coefficient of determination

$$TSS = \sum_{i=1}^{n} (y_{label,i} - \bar{y})^2 \tag{9}$$

$$RSS = \sum_{i=1}^{n} (y_{pred,i} - y_{label,i})^2 \tag{10}$$

Finally we define the Explained sum of squares ESS :

$$ESS = \sum_{i=1}^{n} (y_{pred,i} - \bar{y})^2 \tag{11}$$

Then if the predicitons are linear, then

$$TSS = ESS + RSS \tag{12}$$

Scikit metrics

https://scikit-learn.org/stable/modules/model_evaluation.html

Binary classification

We now review some metrics for binary classification problems.

Accuracy

Most common scoring : accuracy.

$$\frac{\text{number of correct predictions}}{\text{number of samples}} \tag{13}$$

Precision and recall

Precision : "Quand tu dis que c'est positif, c'est positif".

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{14}$$

Recall / sensitivity (rappel) : "Quand c'est positif, tu dis que c'est positif".

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{15}$$

See also : specificity and 1-specificity.

F score

Also called $F1$. It quantifies the tradeoff between precision and recall.

$$F1 = 2 \times \frac{\text{sensitivity} \times \text{precision}}{\text{sensitivity} + \text{precision}} \tag{16}$$

$$F \in [0, 1] \tag{17}$$

`https://en.wikipedia.org/wiki/F-score`

Binary classification

Standard classifiers such as logistic regression and support vector machines are **binary**.
However, sometimes $|\mathcal{Y}| = p > 2$. In these situations, several binary classifiers are aggregated in order to perform the classification.

- one-vs-rest scheme : learns a binary classifier per class. At prediction time, select the class with highest score (there is often some form of confidence score associated with a binary classifier)
- one-vs-one scheme : leanrs as many binary classifiers as there are pairs of classes.

Number of classifiers

We assume $|\mathcal{Y}| = p$.

Exercice 2 : How many classifiers need to be built :

- with the one-vs-rest scheme ?
- with the one-vs-one scheme ?

Number of classifiers

- one-vs-rest is the standard approach.
- one-vs-one need to build a number of classifiers that is quadratic in p, ($\mathcal{O}(p^2)$).
- However one-vs-one might still be useful since less samples are used by each binary classifiers, since we only need the samples from the two selected classes. If the complexity of the classifier scales badly with the umber of samples n (like for kernel methods), one-vs-one might be faster.

Scikit

Scikit has a builtin implementation of both schemes.
`https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html`

Softmax

It is also possible to directly learn p real outputs and predict the maximum. But during training, we need something that we can differentiate.
The softmax approach applies the softmax function to the p outputs, and we train the model to have a softmaxed output close to a discrete disctribution.
`https://fr.wikipedia.org/wiki/Fonction_softmax`

Confusion matrix

https://en.wikipedia.org/wiki/Confusion_matrix
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

Classification report

It is also possible to define precision, recall (and thus F1) for each class. In scikit, classification report prints these quantities for each class

`https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html`

Multi-class vs multi-label

Don't mix multi-class problems and multi-label problems.

- multi-class : several output classes are possible
- mutli-label : we have to make several predictions for each input

A problem can be both multi-class and multi-label.

Hyperparameters (summary of TP3)

All learning algorithms have hyperparameters. Examples :

- regularization parameter
- learning rate schedule
- kernel widths
- tree depth for cart
- number of trees for random forest

Hyperparameter tuning

Sometimes, we have theoretical results that guarantee that a hyperparameter value is a good choice (e.g. the learning rates for GD, SGD, SAG).

However :

- often, these parameters values depend on constants that are problem-dependent and sometimes not available :
  - variance σ^2
  - smoothness constant L
- we may not have a theoretical result at all (true for some aspects of deep learning)
- some values of the hyperparameter might work **better** than the theoretical value.

Hyperparameter tuning

Conclusion : it is most often necessary to experiment and test in order to find relevant hyperparameters.

Train / validation / test sets

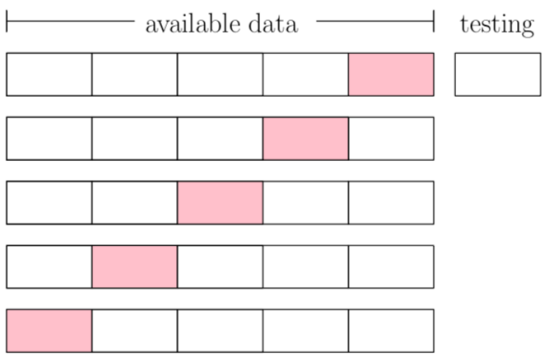
The dataset can be split into 3 parts :

- train set : used to optimize each model
- validation set : used to compute a validation error of each model (error on this dataset). The model with lowest validation error can be chosen.
- test set : used to test the final model. We can not use it for validation (choice of the hyperparameters) : otherwise the estimation of the test error becomes biased.

Problem : there might be a high variability in the validation procedure. The found hyperparameters might depend a lot on the initial choice of the validation set.

Cross-validation

Cross-validation validation is another method that allows the use of more training data. The train set is split in k folds (often 5 of 10), and k validation errors are computed (one for each fold). The model with the lowest average validation error is chosen, **and then** trained on the whole train set.



Due the the higher number of computations, cross-validation might be slower than standard train/validation/test split.

Choice of the set of possible HPs

Grid search is a method for testing hyper parameters (exhaustive search among a fiven list of values).

- grid search : exhaustive
- random search, successive halving.
- bayesian optimization (optuna, skopt)

Learning curves

https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html

Many model selection methods exist :

https://scikit-learn.org/stable/model_selection.html

Unsupervised learning

From a number of samples x_i, you want to retrieve information on their structure : **modelisation**.

Unsupervised learning

From a number of samples x_i, you want to retrieve information on their structure : **modelisation**. The three main unsupervised learning problems are :

- clustering
- density estimation
- dimensionality reduction

Clustering

Clustering consists in partitioning the data. $\forall i, x_i \in \mathcal{X}^n$.

$$D_n = \{(x_i)_{i \in [1,\dots,n]}\} \tag{18}$$

Clustering

Clustering consists in partitioning the data. $\forall i, x_i \in \mathcal{X}^n$.

$$D_n = \{(x_i)_{i \in [1,\dots,n]}\} \tag{19}$$

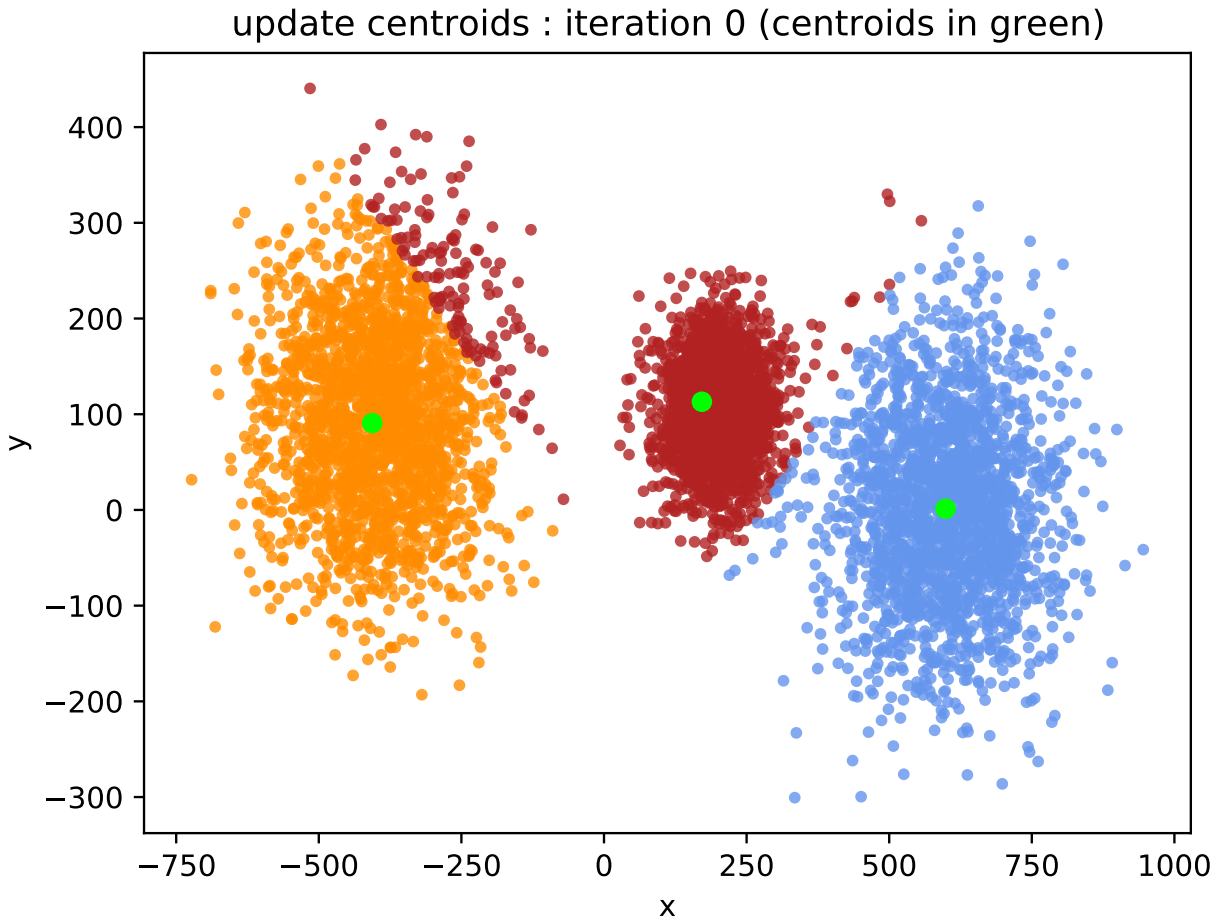
A **partition** is a set of K subsets $A_k \subset D_n$, such that

- $$\cup_{k \in [1,\dots,K]} A_k = D_n \tag{20}$$
- $$\forall k \neq k', A_k \cap A_{k'} = \emptyset \tag{21}$$

Partitions

- **Example 1 :** A is the set of even integers, B the set ot odd integers. Is (A, B) a partition of $\mathbb{N}$?
- **Example 2 :** C is the set of multiples of 2, D the set of multiples of 3. Is (C, D) a partition of $\mathbb{N}$?

Example : partition of data



Applications of clustering

Example applications :

- spam filtering [Sharma and Rastogi, 2014,]
- fake news identification
  [Hosseinimotlagh and Papalexakis, 2018,]
- marketing and sales
- document analysis [Zhao and Karypis, 2002,]
- traffic classificaiton [Woo et al., 2007,]

Some of these applications can be considered to be semi-supervised learning.

Vector quantization

Vector quantization consists in computing **prototypes** $\Omega = (\omega_k)_{k\in[1,\dots,K]} \in \mathcal{X}^K$ that represent the data well.
This implies that a **metric** is defined on $\mathcal{X}$.
Most often, this is interesting / useful if $K << n$.

Voronoï subsets

We assume a loss (we can think of it as a distance here) L is defined on $\mathcal{X}$. The **Voronoï subset** of ω is defined as

$$V(\omega) = \{x \in D_n, \arg\min_{\omega' \in \Omega} L(\omega', x) = \omega\} \tag{22}$$

- We assume that arg min returns one single element.
- The Voronoï subsets form a partition of D_n.

Distortion

To measure the quality of a Voronoï partition, we introduce the **distortion** $R(\Omega)$.
For each x, we note $h_\Omega(x) = \arg\min_{\omega' \in \Omega} L(\omega', x)$.

$$R(\Omega) = \frac{1}{n} \sum_{i=1}^{n} L(x_i, h_\Omega(x_i)) \tag{23}$$

Distortion

For each x, we note $h_\Omega(x) = \arg\min_{\omega' \in \Omega} L(\omega', x)$.

$$\begin{aligned} R(\Omega) &= \frac{1}{n} \sum_{i=1}^{n} L(x_i, h_\Omega(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} \sum_{x \in V(\omega)} L(x_i, h_\Omega(x_i)) \\ &= \frac{1}{n} \sum_{\omega \in \Omega} V_\Omega(\omega) \end{aligned} \tag{24}$$

with

$$V_\Omega(\omega) = \sum_{x \in V(\omega)} L(x_i, h_\Omega(x_i)) \tag{25}$$

Minimum of distortion

We want to find the prototypes for which the distortion is **minimal**.

- The set of prototypes minimizing distorsion might not be unique.
- We need to tune K (number of prototypes).

Vector quantization techniques

- K-means
- Growing neural gas (GNG)
- Self-organizing maps

K-means clustering

- $\mathcal{X} = \mathbb{R}^d$.
- $L(x, y) = ||x - y||^2$.

Objective function

With

- $\Omega = \{\omega_1, \ldots, \omega_K\} \in \mathbb{R}^{K,d}$.
- $z_i^k = 1$ if x_i is assigned to ω_k, $z_i^k = 0$ otherwise. $z = (z_i^k) \in \mathbb{R}^{n,K}$.

we define the objective function

$$J(\Omega, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k ||x_i - \omega_k||^2 \tag{26}$$

It is also called the **inertia.**

K-means algorithm

Result: $\Omega \in \mathbb{R}^{K,d}$
$\Omega \leftarrow$ Random initialization;
$z = M$ where M is the $n \times K$ matrix with 0's;
while *Convergence criteria is not satisfied* **do**
  a] Greedily minimize J with respect to z;
  b] Minimize J with respect to Ω;
end
return Ω

Algorithm 1: K-means (Loyd algorithm)

Stopping criterion

To stop the algorithm, the norm of the difference between Ω_t and Ω_{t+1} must be smaller than a given tolerance. (e.g. $1e^{-4}$). Here it is a norm between matrix (Frobenius norm) :

$$||A||_F = \sqrt{\sum_{i=1}^{n} A_{ij}^2} \tag{27}$$

Minimization

We focus on step b]. How can we minimize J with respect to Ω ?

Minimization

Exercice 3 : **Convexity :**
Show that $J(\Omega, z)$ is convex with respect to Ω.

$$J(\Omega, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k ||x_i - \omega_k||^2 \tag{28}$$

Hence, to minimize J with respect to Ω, we just need to cancel the gradient.

Minimization

Exercice 4 : **Gradient :**

Compute the gradient of $J(\Omega, z)$ with respect to Ω and deduce the minimizer Ω^*.

- z is fixed
- we can see Ω has a vector of $\mathbb{R}^{Kd}$.

Convexity

Is $J(\Omega, z)$ convex in z ?

Convexity

Is $J(\Omega, z)$ convex in z ?
No, as z is not even defined on a convex set, so convexity can not be defined anyways !
Hence, the function might have **local minima**.

Suboptimal clustering

Exercice 4 : **Local minima** : propose a setting (dataset, initialization) for which the algorithm outputs a bad set of centroids.

Suboptimal clustering

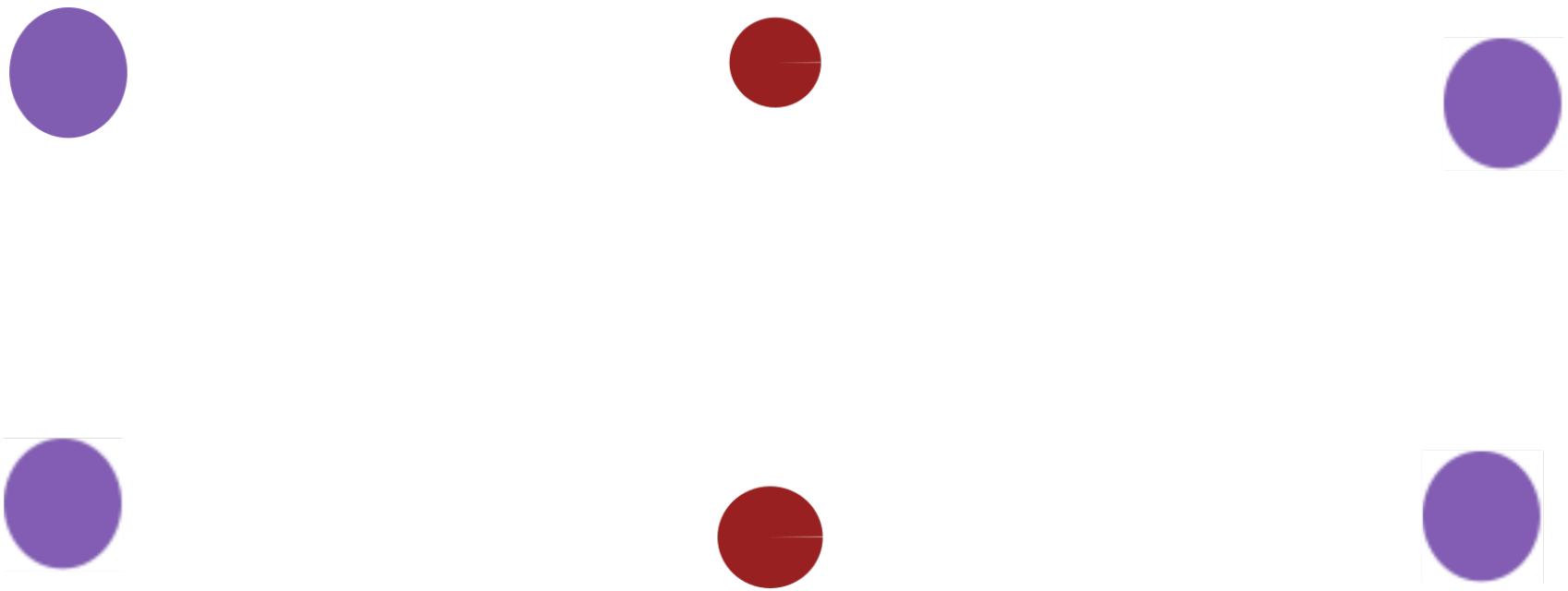


Figure – Initialization of centroids in red.

Random initialization

Hence, the result of K-means strongly depends on the initial position of the centroids.
A common approach is to restart the algorithm several times and select the result with lowest inertia.

Drawbacks of inertia minimization

K-means is based on the minimization of the inertia and hence on the euclidean distance. **However**, in some contexts, the euclidean distance is not the adapted metric.

`https://scikit-learn.org/stable/modules/clustering.html`

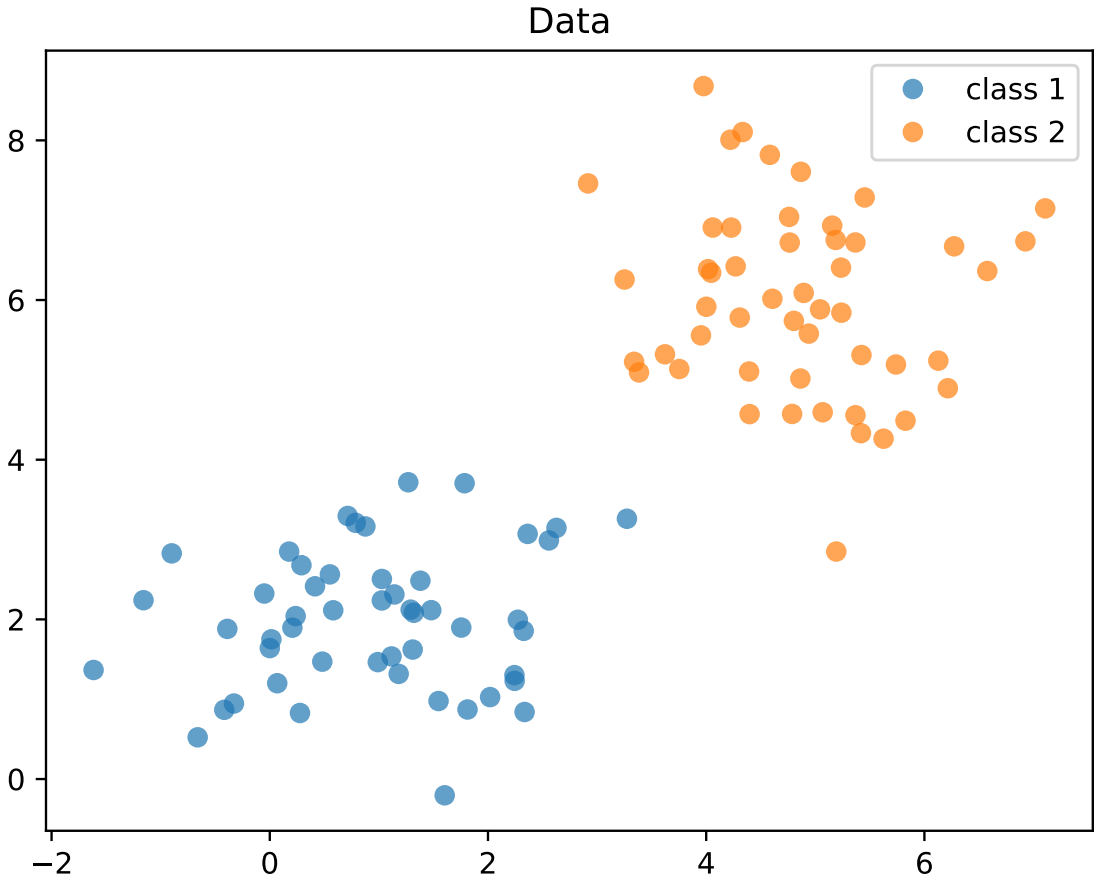


Figure – Linearly separable data

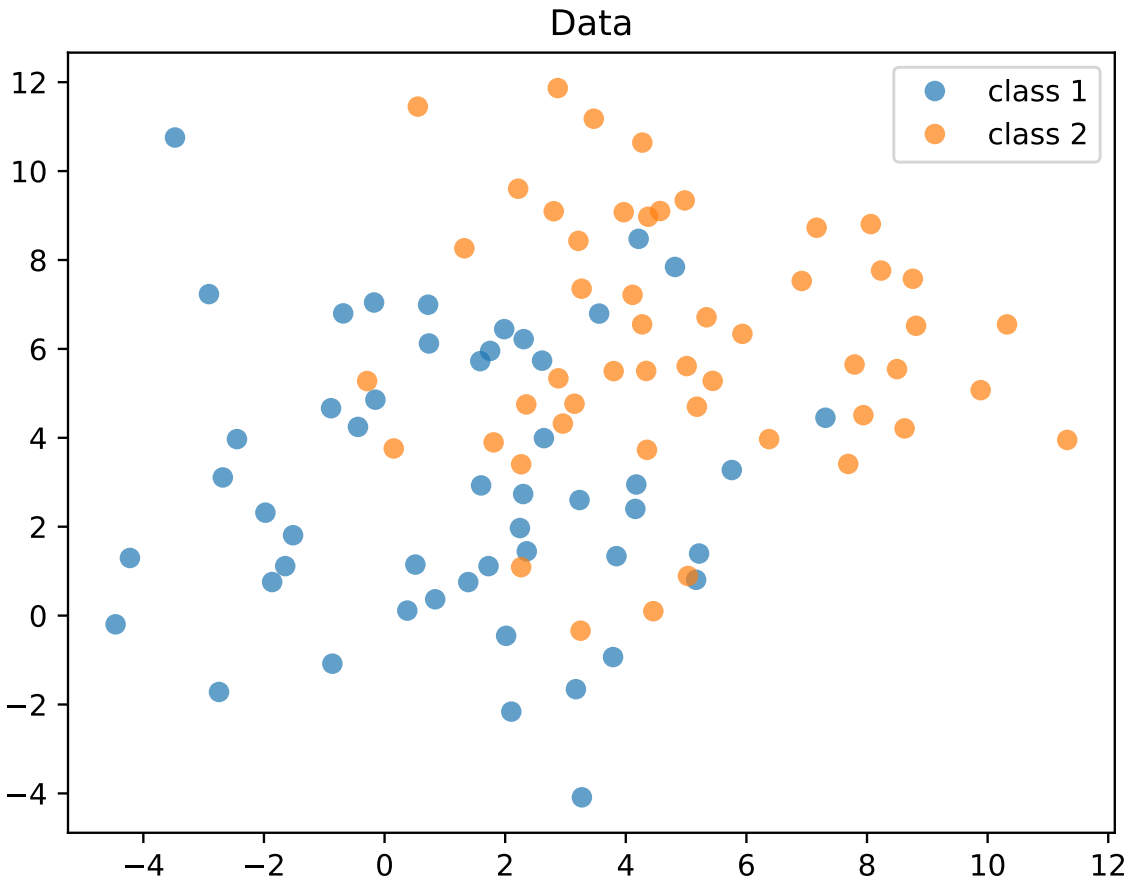


Figure – Non linearly-separable data

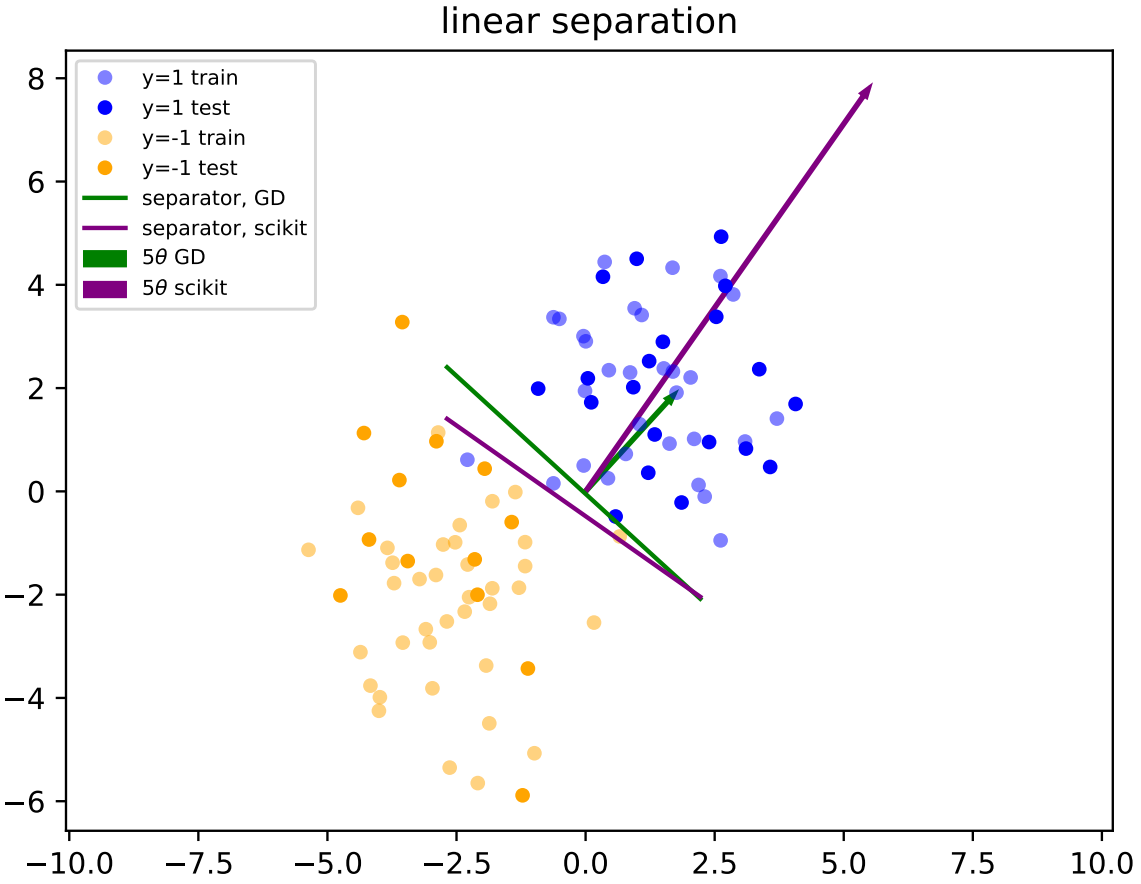


Figure – Linear separator

Linear separator

- $\mathcal{X} = \mathbb{R}^d$
- $\mathcal{Y} = \{-1, 1\}$

Equation of a linear separator

$$\langle w, x \rangle + b = 0 \tag{29}$$

- $w \in \mathbb{R}^d$
- $x \in \mathbb{R}^d$
- $b \in \mathbb{R}$

Notation :

$$h_{w,b}(x) = \langle w, x \rangle + b \tag{30}$$

Affine subspace

$$H = \{x \in \mathbb{R}^d, \langle w, x \rangle + b = 0\} \tag{31}$$

is an affine subspace.

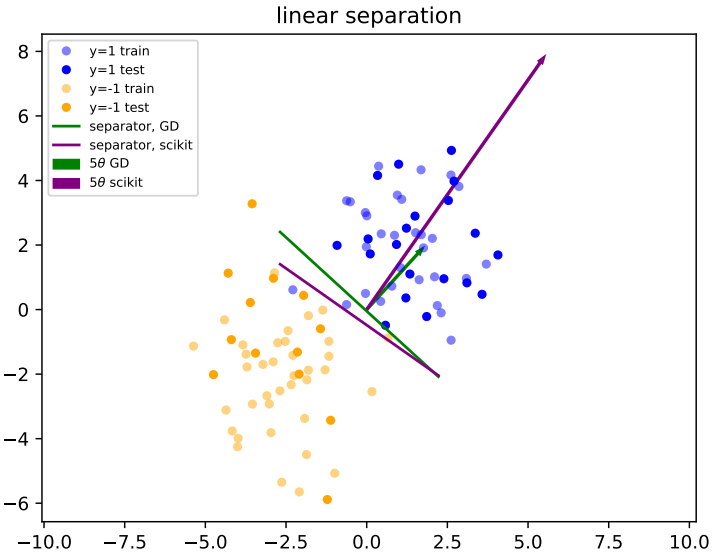
Any vector $x \in \mathbb{R}^d$ can uniquely be decomposed as

$$x = \lambda_w^x \frac{w}{||w||} + x_{w^\perp} \tag{32}$$

with $x_{w^\perp} \in \text{vect}(w)^\perp$. $x \in H$ if and only if

$$\begin{aligned} \langle w, x \rangle + b = 0 \\ \Leftrightarrow \langle w, \lambda_w^x \frac{w}{||w||} + x_{w^\perp} \rangle + b = 0 \\ \Leftrightarrow \langle w, \lambda_w^x \frac{w}{||w||} \rangle + b = 0 \\ \Leftrightarrow \lambda_w^x ||w|| + b = 0 \\ \Leftrightarrow \lambda_w^x = \frac{-b}{||w||} \end{aligned} \tag{33}$$

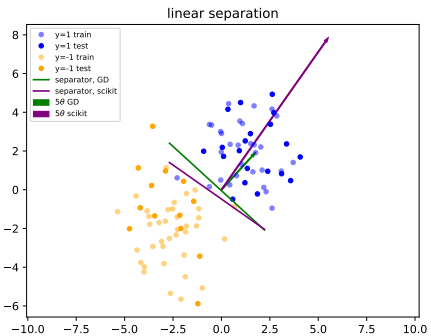
We first consider a linearly separable situation.



We recall the definition $h_{w,b}(x) = \langle w, x \rangle + b$. We look for separators that satisfy :

- $\forall x_i$ such that $y_i = 1$, $h_{w,b}(x) \geq 0$
- $\forall x_i$ such that $y_i = -1$, $h_{w,b}(x) \leq 0$

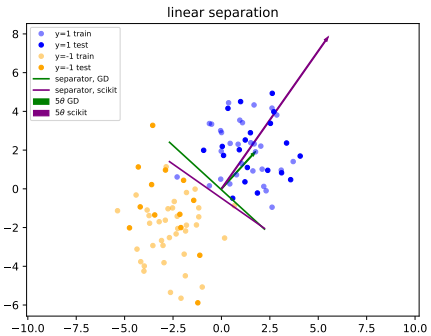
We first consider a linearly separable situation.



We note $h_{w,b}(x) = \langle w, x \rangle + b$. We look for separators that satisfy :

- $\forall x_i$ such that $y_i = 1$, $h_{w,b}(x) \geq 0$
- $\forall x_i$ such that $y_i = -1$, $h_{w,b}(x) \leq 0$

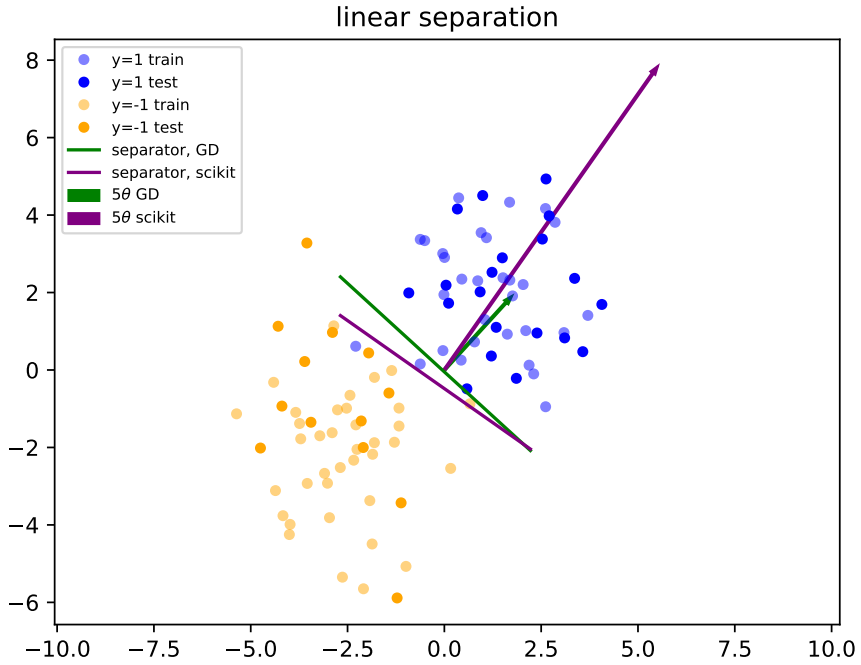
However, there exists an infinite number of such parameters. How could we choose the best one ?



- $\forall x_i$ such that $y_i = 1$, $h_{w,b}(x) \geq 0$
- $\forall x_i$ such that $y_i = -1$, $h_{w,b}(x) \leq 0$

The **margin** is the distance from H to the dataset. We look for the separator with the largest margin, leading to **Support vector classification (SVC)**.

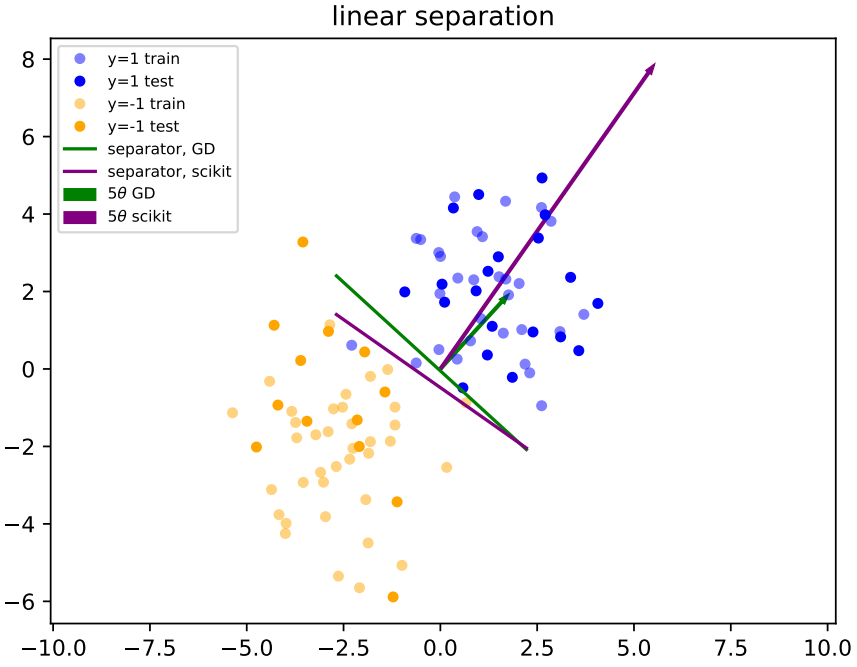
Margin



Let x be a point such that $h_{w,b}(x) = \langle w, x \rangle + b = c$, with $c \in \mathbb{R}$.

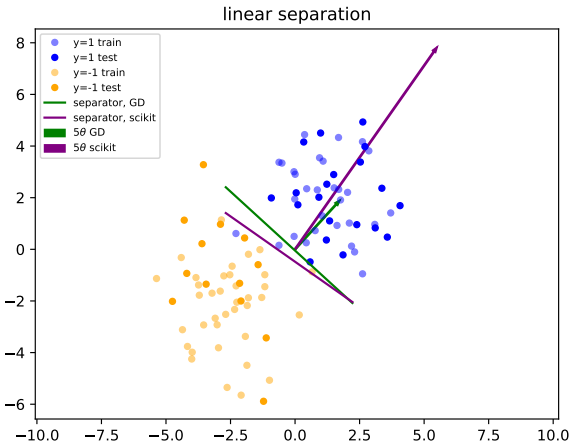
Exercice 5 : Compute the distance from x to H.

Margin



Let x be a point such that $h_{w,b}(x) = \langle w, x \rangle + b = c$, with $c \in \mathbb{R}$.
The distance is $\frac{|c|}{||w||}$.

Support vectors

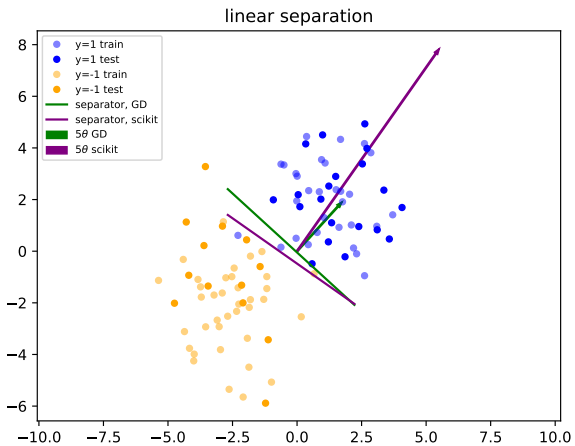


The **support vectors** are the vectors such that $|h_{w,b}(x)|$ is minimal among the dataset.

- the margin M is the distance from H to these vectors.
- if H is the optimal separator, there has to be a vector x_- and x_+ on each side, such that

$$M = d(x_-, H) = d(x_+, H) \tag{34}$$

Support vectors



Exercice 6 : Show that if H is optimal, then

$$M = d(x_-, H) = d(x_+, H) \tag{35}$$

Rescaling

Important remark : multiplying w and b by a constant $\lambda \neq 0$ does not change H, as :

$$\begin{aligned} &\langle \lambda w, x \rangle + \lambda b = 0 \\ &\Leftrightarrow \lambda(\langle w, x \rangle + b) = 0 \\ &\Leftrightarrow \langle w, x \rangle + b = 0 \end{aligned} \tag{36}$$

Rescaling

Important remark : multiplying w and b by a constant $\lambda \neq 0$ does not change H.
If the support vector x is such that $h_{w,b}(x) = c$, we have seen that the margin is

$$\frac{|c|}{||w||} \tag{37}$$

When looking for the optimal H, we can impose, without loss of generality, that $|c| = 1$.
This means that we look for w with minimal norm, such that H separates the data (since the margin is $\frac{1}{||w||}$).

Optimization problem

We can now formulate the optimization problem.

$$\arg\min_{w,b} \frac{1}{2}\langle w, w\rangle \tag{38}$$

subject to :

$$\forall i \in [1, n], y_i(\langle w, x_i\rangle + b) \geq 1 \tag{39}$$

Slack variables

When the dataset is not linearly separable, the approach is to authorize some of the samples to have a margin smaller that 1. This means relaxing the constraint, from

$$y_i(\langle w, x_i \rangle + b) \geq 1 \tag{40}$$

to

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \tag{41}$$

The ξ are called the *slack variables*, they are ≥ 0. The smaller the slack variabes, the better.

Optimization problem

In the general case, the optimization problem is :

$$\arg\min_{w,b,\xi} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n}\xi_i \tag{42}$$

subject to :

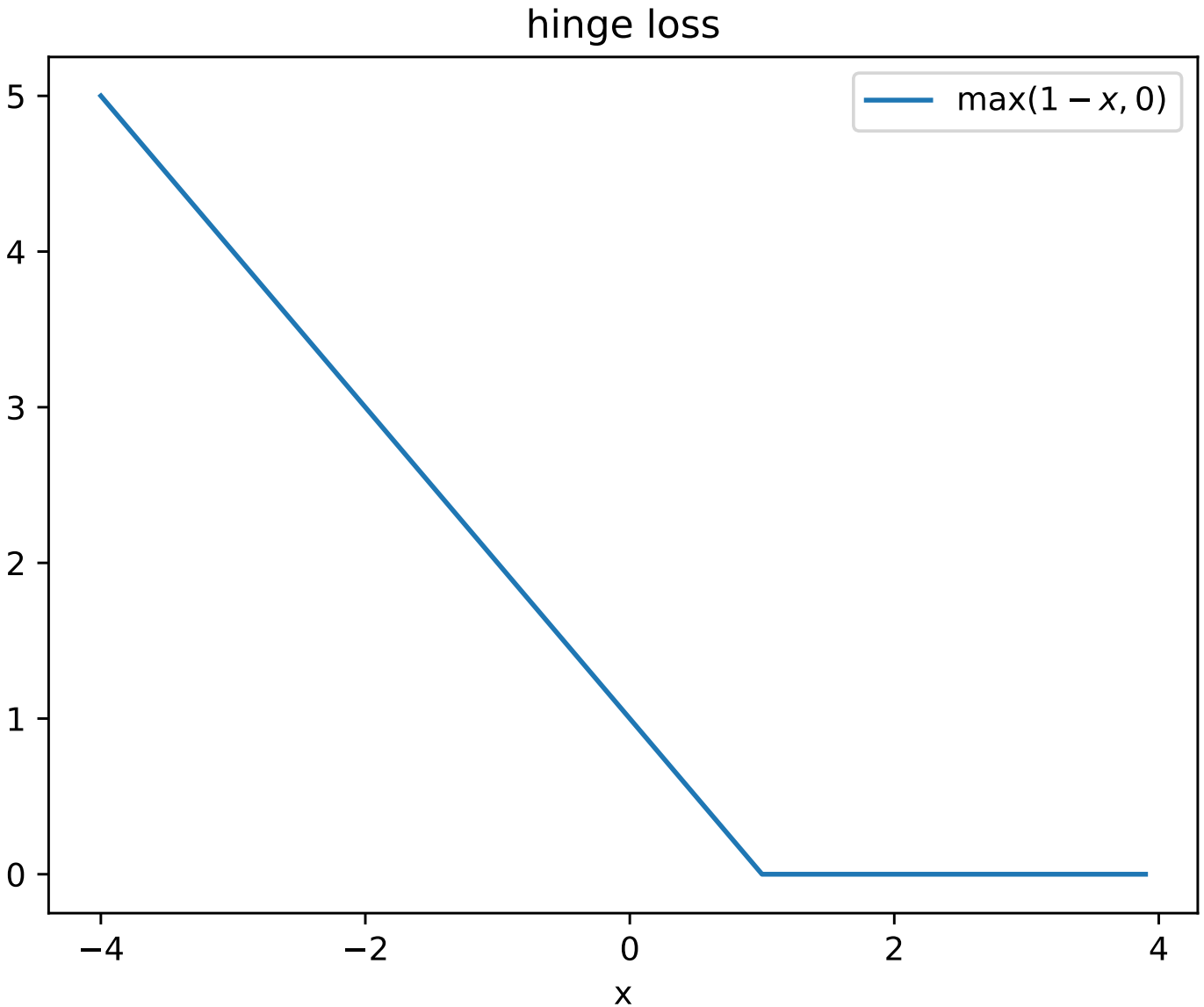
$$\forall i \in [1, n], y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i \tag{43}$$

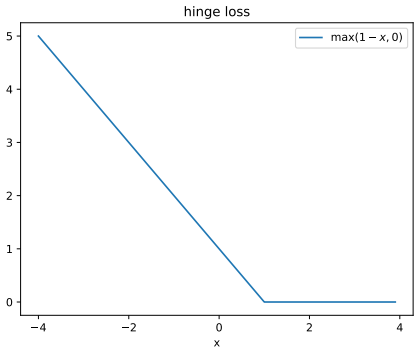
and

$$\forall i \in [1, n], \xi_i \geq 0 \tag{44}$$

Margin vs ERM

The margin maximisation seems to differ from empirical risk minimization (ERM), which we have studied earlier.
However, with a specific loss function, we an show that margin maximisation is in fact an ERM.


hinge loss
max(1 − x, 0)
5
4
3
2
1
0
−4
−2
0
2
4
x



- estimation : $h(x) = \langle w, x \rangle + b$
- label : $y \in \{-1, 1\}$

Hinge loss :

$$L_{\text{hinge}}(h(x), y) = \max(0, 1 - yh(x)) \tag{45}$$

The hinge loss can be seen as an approximation of the binary loss.

Problem reformulation

We recall the constraints on ξ

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \tag{46}$$

and

$$\xi_i \geq 0 \tag{47}$$

Equivalently,

$$\xi_i \geq max\big(0, 1 - y_i(\langle w, x_i \rangle + b)\big) \tag{48}$$

Problem reformulation

The slack variables should be minimal. Hence, we can write that for the optimal solution, the inequality is in fact an equality ;

$$\xi_i = max\big(0, 1 - y_i(\langle w, x_i \rangle + b)\big) \tag{49}$$

Problem reformulation

Finally, we can rewrite the problem as

$$\arg\min_{w,b} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n} max\big(0, 1 - y_i(\langle w, x_i\rangle + b)\big) \tag{50}$$

or equivalently

$$\arg\min_{w,b} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{n} L_{\text{hinge}}(h(x_i)), y_i) \tag{51}$$

Which is an ERM problem with a $L2$ regularization.

References I


Hosseinimotlagh, S. and Papalexakis, E. E. (2018).
Unsupervised content-based identification of fake news articles with tensor decomposition ensembles.
Proceedings of the WSDM MIS2 : Misinformation and Misbehavior Mining on the Web Workshop, pages 1–8.


Sharma, A. and Rastogi, V. (2014).
Spam Filtering using K mean Clustering with Local Feature Selection Classifier.
International Journal of Computer Applications, 108(10) :35–39.

References II


Woo, D. M., Park, D. C., Song, Y. S., Nguyen, Q. D., and Tran, Q. D. N. (2007).
Terrain classification using clustering algorithms.
Proceedings - Third International Conference on Natural Computation, ICNC 2007, 1 :315–319.


Zhao, Y. and Karypis, G. (2002).
Evaluation of hierarchical clustering algorithms for document datasets.
International Conference on Information and Knowledge Management, Proceedings, (August 2002) :515–524.